

Arnold Willemer

```
while (true)
  name = getname();
  festgemacht = getzeit();
  getzeit();
  wunschfunktion();
  while (true)
    name = getname();
    festgemacht = getzeit();
    getzeit();
    wunschfunktion();
```

GARANTIIERT  
KEIN  
LEHRBUCH!



# Coding for Fun

mit C++



- Das nächste Level für C++-Programmierer!
- Spannende Programme verstehen und selbst entwickeln
- Nachfahrt, Frosch & Eichhörnchen, Eliza, Labyrinth, Navis, Mondlandung u. v. m.

Auf CD: C++-Compiler, Tools und natürlich  
alle Code-Beispiele des Buchs

Galileo Computing

mit C++

Coding for Fun

Willemer



1512



# Auf einen Blick

1	C++ – Das Porträt .....	13
2	Der Computer im Dialog .....	21
3	Pfadfinder .....	45
4	Spiele mit der Physik .....	63
5	Regionales C++ .....	87
6	Labyrinth .....	107
7	Sprunghaft .....	137
8	Esoterische Software .....	151
9	Die dritte Dimension .....	167
10	Shorts im Nebel .....	207
11	Achtung Baustelle: Einsturzgefahr! .....	241
12	Der Frosch und das Eichhörnchen .....	255
13	Musik ist mit Wellen verbunden .....	277
A	KDevelop .....	311
B	Bloodshed Dev C++ .....	315
C	Installation von wxWidgets .....	319
D	Installation PortAudio .....	327
E	Computer-Oldies .....	331

# Inhaltsverzeichnis

Geleitwort des Fachgutachters .....	11
<b>1 C++ – Das Porträt .....</b>	<b>13</b>
1.1 Buchkonzept .....	13
1.2 Blick zurück .....	14
1.3 Spaß mit C++ .....	18
<b>2 Der Computer im Dialog .....</b>	<b>21</b>
2.1 Eliza .....	21
2.1.1 Unsere kleine Eliza .....	24
2.1.2 Die Datenmodellierung .....	24
2.1.3 Der Programmablauf .....	28
2.1.4 Ein Sitzungsprotokoll .....	34
2.1.5 Was denn noch? .....	35
2.2 Tiere raten .....	36
2.2.1 Die Spielidee .....	36
2.2.2 Datenstruktur Binärbaum .....	37
2.2.3 Eine Rekursion muss her .....	38
2.2.4 Ein Spielprotokoll .....	42
2.2.5 Was denn noch? .....	43
<b>3 Pfadfinder .....</b>	<b>45</b>
3.1 Das Programm sucht die kürzeste Verbindung .....	47
3.2 Datenmodellierung .....	53
3.2.1 Knoten .....	53
3.2.2 Kanten .....	54
3.2.3 Adjazenzmatrix .....	54
3.3 Einlesen der Graphen .....	57
3.4 Das Hauptprogramm .....	58
3.5 Betrachtungen über die Straßendaten .....	59
3.6 Navigationssysteme .....	61
3.7 Was denn noch? .....	61

<b>4</b>	<b>Spiele mit der Physik</b>	<b>63</b>
4.1	Die Mondlandung	64
4.1.1	Ein wenig Physik	64
4.1.2	Die Tabellenkalkulation	66
4.1.3	Das Ganze in C++	69
4.1.4	Die Raumfähre	70
4.1.5	Das Hauptprogramm	72
4.1.6	Was denn noch?	73
4.2	Der Kaugummi und der schiefe Wurf	74
4.2.1	Die Formel	75
4.2.2	Hilfe von der Tabellenkalkulation	76
4.2.3	Spielfeld aufbauen	78
4.2.4	Spielablauf	81
4.2.5	Was denn noch?	85
<b>5</b>	<b>Regionales C++</b>	<b>87</b>
5.1	Hessisch?	89
5.1.1	Ei, wie spricht mer des?	90
5.1.2	Die hessische Tabelle	90
5.1.3	Was hinten herauskommt	91
5.2	Mer programmiere in Hessisch	92
5.2.1	Tabellengesteuert	93
5.2.2	Erzeugen der Header-Datei	95
5.3	Mer lese Listings	96
5.3.1	Zugriff auf die Übersetzungstabelle	97
5.3.2	Durchlesen des Quelltextes	97
5.3.3	String gegen Zeiger	99
<b>6</b>	<b>Labyrinth</b>	<b>107</b>
6.1	Labyrinthmodell	107
6.1.1	Vorgehensweise	108
6.1.2	Datenstrukturen	110
6.1.3	Zufälliger Richtungswechsel	114
6.1.4	Selbstentzünder	115
6.1.5	Ausgang erzeugen	119
6.1.6	Ausgabe auf dem Terminal	120
6.1.7	Aufrufparameter des Hauptprogramms	122

6.2	Wandelgang .....	123
6.2.1	Wände im Raum .....	124
6.2.2	Bewegung durch das Labyrinth .....	127
6.2.3	Mitmachaktion: eine Multifunktionsroutine .....	129
6.2.4	ASCII-Panel .....	131
6.3	Was denn noch? .....	135
<b>7</b>	<b>Sprunghaft .....</b>	<b>137</b>
7.1	Grafische Oberflächen .....	137
7.2	Das Superprogramm .....	139
7.3	Ereignisorientiert .....	140
7.3.1	Der Grundrahmen .....	140
7.3.2	Fensterelementebau .....	143
7.3.3	Ereignisbehandlung .....	145
7.4	Was denn noch? .....	149
<b>8</b>	<b>Esoterische Software .....</b>	<b>151</b>
8.1	Strahlungen und Energien .....	152
8.1.1	Messen der »Good Vibrations« .....	152
8.1.2	Sonnenstrahlen sind nicht blau .....	153
8.1.3	Schwingungen mechanisch messen .....	154
8.2	Programmieren .....	154
8.2.1	Fensterklassen .....	155
8.2.2	Ereignisse .....	156
8.2.3	Ellipsen .....	158
8.2.4	Menüspielereien .....	162
8.3	Was denn noch? .....	163
<b>9</b>	<b>Die dritte Dimension .....</b>	<b>167</b>
9.1	Fluglandung .....	167
9.1.1	Faszination Fliegen .....	167
9.1.2	Flugsimulator-Computer contra Jackintosh .....	168
9.1.3	Apple-Fluglandung .....	170
9.1.4	Ein Blick ins Listing .....	170
9.1.5	Ein paar störende Fakten .....	171
9.1.6	Simulation des Fluges .....	172
9.1.7	Die Ansicht der Landebahn .....	174
9.1.8	Die grafische Oberfläche .....	177

9.1.9	Besinnung .....	182
9.1.10	Eine Portierung nach Win32 .....	183
9.1.11	Was denn noch? .....	190
9.2	Nightdriver .....	191
9.2.1	Modellierung des Straßenverlaufs .....	192
9.2.2	Grafische Umsetzung .....	193
9.2.3	Die Anwendungsklasse .....	200
9.2.4	Was denn noch? .....	204
<b>10</b>	<b>Shorts im Nebel .....</b>	<b>207</b>
10.1	Spielregeln .....	208
10.2	Datenmodellierung .....	208
10.2.1	Ein Schiff .....	209
10.2.2	Das Wasser .....	209
10.3	Die Spielimplementierung .....	210
10.4	Konsolenversion .....	214
10.5	Etwas Komfort .....	216
10.6	Nun alles im Fenster .....	217
10.7	Grafik kann mehr .....	223
10.7.1	Grundaufbau des Feldes .....	224
10.7.2	Die Diagonalen als Knobelspiel .....	224
10.7.3	Präzise Maussteuerung .....	227
10.7.4	Menü mit Haken .....	230
10.7.5	Image ändern .....	231
10.7.6	Spielergebnisse zeigen .....	232
10.7.7	Die Peilung rotiert .....	234
10.7.8	Flackernden Neuaufbau unterbinden .....	236
10.8	Portabilität .....	237
10.9	Was denn noch? .....	239
<b>11</b>	<b>Achtung Baustelle: Einsturzgefahr! .....</b>	<b>241</b>
11.1	Unsauberes Konzept .....	243
11.2	Chaotische Implementierung .....	244
11.3	Außen hui .....	249
11.4	Was denn noch? .....	254

<b>12 Der Frosch und das Eichhörnchen</b>	<b>255</b>
12.1 Rahmenprogramm	256
12.2 Diashow	261
12.2.1 Hintergrundwechsler	261
12.3 Es bewegt sich etwas	263
12.4 Aufstellung der Figuren	266
12.5 Die Zeit setzt in Bewegung	268
12.6 Malerei	270
12.7 Angetastet	273
12.8 Spielende	274
12.9 Was denn noch?	275
<b>13 Musik ist mit Wellen verbunden</b>	<b>277</b>
13.1 Musik digitalisieren	279
13.2 Musik und Geräusche abspielen	280
13.3 Synthesizer	283
13.3.1 Das Sägezahn-drama	284
13.3.2 Gemeinheiten des Soundsystems	288
13.3.3 Die weiche Welle	289
13.4 Achtung Aufnahme!	290
13.5 Stimmungskanone	296
13.5.1 Die Daten	297
13.5.2 Das Hauptprogramm	298
13.5.3 Was denn noch?	300
13.6 Der gute Ruf des Spielers	302
13.6.1 Akustischer Hau-den-Lukas	303
13.6.2 Die Umsetzung	303
13.6.3 Was denn noch?	306
<b>Anhang</b>	<b>309</b>
A KDevelop	311
A.1 Neues Projekt	312
A.2 Kompilieren und starten	313
A.3 Weitere Möglichkeiten	314
B Bloodshed Dev C++	315
B.1 Installation	315
B.2 Ein Projekt anlegen	316
B.3 Übersetzen und starten	317

C	Installation von wxWidgets .....	319
C.1	Installation unter KDevelop .....	319
C.2	Installation unter Bloodshed Dev C++ .....	320
C.2.1	Buch-CD .....	320
C.2.2	Internetinstallation .....	320
C.2.3	Erzeugen einer Beispielapplikation .....	321
C.2.4	Buchprojekte .....	322
C.3	Installation unter Visual Studio C++ .....	322
C.4	Installation unter Macintosh .....	324
D	Installation PortAudio .....	327
D.1	Unter Linux .....	327
D.2	Unter Windows .....	327
D.2.1	Compileraufruf .....	328
D.2.2	Selbstbau der Bibliothek .....	329
D.3	Lizenz .....	330
E	Computer-Oldies .....	331
E.1	Apple ][ .....	331
E.2	Atari 400/800 .....	335
E.3	Sinclair ZX80 .....	337
E.4	Commodore C-64 .....	338
E.5	Schneider/Amstrad CPC .....	340
E.6	Epson HX-20 .....	342
E.7	IBM PC .....	344
E.8	Apple Macintosh .....	346
E.9	Atari ST .....	348
E.10	Commodore Amiga .....	350
	Index .....	353



# 1 C++ – Das Porträt

Sie programmieren in C++. Dann sind Sie hier richtig. Sollten Sie diese Sprache dagegen noch nicht beherrschen, will ich nicht ausschließen, dass Sie dieses Buch an der einen oder anderen Stelle etwas unverständlich finden werden. Aber da gibt es Abhilfe. Im gleichen Verlag finden Sie ein gutes Buch namens »Einstieg in C++«. Die Namensgleichheit des Autors ist nicht einmal zufällig. Vielleicht lesen Sie zunächst jenes Buch und dann dieses.

Die Anfänger dürften nun das Buch zugeklappt haben und erst in ein paar Tagen oder Wochen wieder zu uns stoßen, nachdem sie C++ gelernt haben.

Sie als Kenner der Sprache C++ werden in diesem Kapitel die Gelegenheit haben, etwas über die Geschichte und den Charakter von C++ zu erfahren. Sie werden also all das erfahren, was Sie noch nie hören wollten oder längst schon wissen. Daneben werden Sie lesen, wie dieses Buch konzipiert ist. Falls Sie das Buch aber bereits gekauft und bezahlt haben oder wenn Sie lieber überrascht werden wollen, schlage ich vor, Sie überblättern dieses Kapitel und sparen damit Zeit.

Spätestens jetzt müsste ich allein sein und sollte mir die Frage stellen, warum ich dieses Kapitel geschrieben habe, wenn es doch keiner liest.

## 1.1 Buchkonzept

Programmieren ist eine besondere Form des Knobeln. Wer programmiert, löst täglich Denksportaufgaben. Der eine oder andere hat diese Knochelei zu seinem Beruf gemacht. Aber die typischen Aufgaben im Beruf sind meist eher technischer Natur. Und so wie die nichtprogrammierenden Artgenossen das Sudoku in der Tageszeitung lösen, hat der Programmierer hin und wieder Lust, etwas zu programmieren, was einfach nur Spaß macht.

In diesem Buch sind Themen aufgegriffen worden, die sich für solche Knocheleien eignen. Die Kapitel bauen nicht aufeinander auf, sondern können weitgehend in beliebiger Reihenfolge gelesen werden. Sie sind nicht völlig ohne Sinn und Verstand in dieser Reihenfolge angeordnet worden, aber beinahe.

In jedem Kapitel wird zunächst in das Thema eingeführt. Dann wird ein Lösungsansatz vorgestellt und der Quelltext beschrieben. Schließlich werden Ihnen ein paar Ideen aufgezeigt, wie Sie die Programme erweitern können. Ich kann mir vorstellen, dass Sie selbst eigene Ideen haben.

Ich verstehe dieses Buch durchaus als Mitmachbuch. Warum sollte ich den Spaß auch allein gehabt haben. Der Verlag stellt freundlicherweise zu diesem Buch ein Forum bereit: <http://www.galileocomputing.de/forum/gp/forumID-241>. Stellen Sie dort Ihre Lösungen vor. Es wäre nett, von Ihnen zu hören.

Die vorgestellten Programme sind also nicht komplett und fertig, sondern eher Appetithappen. Sie erheben auch nicht den Anspruch, vorbildliches C++ zu predigen. Und darum werden Sie vielleicht an der einen oder anderen Stelle bemängeln, dass die Fehlerbehandlung fehlt, Klassen mit öffentlichen Membervariablen definiert sind und andere Verbrechen gegen die Ideologie der sauberen Programmentwicklung begangen wurden. Der Grund dafür ist, dass die Programme möglichst kurz und übersichtlich sein sollten. Sie sollen Spaß machen, nicht Vorbild sein.

## 1.2 Blick zurück

Wer die Geschichte der Programmiersprache C++ betrachten möchte, sollte ein Fernglas verwenden, schließlich geht es weit zurück in die Vorzeit. Denn damals, kurz vor der Erfindung der Rasierklinge, entwickelten Brian Kernighan und Dennis Ritchie die Programmiersprache C, um das Betriebssystem UNIX nicht in Assembler schreiben zu müssen.

Eine unbekannte Quelle (Bernhard L. Hayes, NetNews-Gruppe) schrieb in einem Artikel unter dem Titel »Erfinder von UNIX und C geben zu: alles Quatsch!«, dass UNIX und C nur ein Aprilscherz gewesen sei.

*In einer Ankündigung, die die Computerindustrie verblüffte, haben Ken Thompson, Dennis Ritchie und Brian Kernighan zugegeben, dass das Betriebssystem Unix und die Programmiersprache C ein raffinierter Aprilscherz sind, der sich über 20 Jahre am Leben erhalten hat.*

*Bei einem Vortrag vor dem letzten UnixWorld-Software-Entwicklungsforum enthüllte Thompson:*

*1969 hatte AT&T gerade die Arbeit am GE/Honeywell/AT&T-Multics-Projekt beendet. Brian und ich experimentierten zu dem Zeitpunkt mit einer frühen Pascal-Version von Professor Niklaus Wirth vom ETH-Laboratorium in der Schweiz und waren beeindruckt von seiner Einfachheit und Mächtigkeit. Dennis hatte »Der Herr der*

*Klinge« gelesen, eine spöttische Parodie auf Tolkiens große Triologie »Der Herr der Ringe«.*

*Im Übermut beschlossen wir, Parodien zur Multics-Umgebung und zu Pascal zu verfassen. Dennis und ich waren für die Betriebssystemumgebung verantwortlich. Wir sahen uns Multics an und entwarfen ein neues System, das so komplex und kryptisch wie möglich sein sollte, um die Frustration der gelegentlichen Nutzer zu maximieren. Wir nannten es Unix, in Anspielung auf Multics, und fanden es auch nicht gewagter als andere Verballhornungen. Danach entwickelten Dennis und Brian eine wirklich »perverse« Pascal-Version namens »A«. Als wir bemerkten, dass einige Leute tatsächlich versuchten, in A zu programmieren, fügten wir schnell einige zusätzliche Fallstricke hinzu und nannten es B, BCPL und schließlich C. Wir hörten damit auf, als wir eine saubere Übersetzung der folgenden Konstruktion erhielten:*

```
for(;P("\n"),R--;P("!"))for(e=C;e--;P("_"+(*u++/8)%2))
```

*Der Gedanke, dass moderne Programmierer eine Sprache benutzen würden, die solch eine Anweisung zuließ, lag jenseits unseres Vorstellungsvermögen. Wir dachten allerdings daran, alles den Sowjets zu verkaufen, um ihren Computerfortschritt 20 Jahre und mehr zu behindern. Unsere Überraschung war groß, als dann AT&T und andere US-Unternehmen tatsächlich begannen, Unix und C zu verwenden! Sie haben 20 weitere Jahre gebraucht, genügend Erfahrungen zu sammeln, um einige bedeutungslose Programme in C zu entwickeln und das mit einer Parodie auf die Technik der 60er-Jahre! Dennoch sind wir beeindruckt von der Hartnäckigkeit (falls nicht doch Gemeinsinn) des gewöhnlichen Unix- und C-Anwenders. Jedenfalls haben Brian, Dennis und ich in den letzten Jahren nur in Pascal auf einem Apple Macintosh programmiert, und wir fühlen uns echt schuldig an dem Chaos, der Verwirrung und dem wirklich schlechten Programmierstil, der von unserem verrückten Einfall vor so langer Zeit ausging.*

*Namenhafte Unix- und C-Anbieter und -Benutzer, einschließlich AT&T, Microsoft, Hewlett-Packard, GTE, NCR und DEC haben vorläufig jede Stellungnahme abgelehnt. Borland International meinte, sie hätten diesen Verdacht schon seit Jahren gehegt und würden nun dazu übergehen, ihre Pascal-Produkte zu verbessern und weitere Bemühungen um die C-Entwicklung stoppen. Ein IBM-Sprecher brach in unkontrolliertes Gelächter aus.*

Den Artikel finden Sie in verschiedenen Varianten unter den Stichworten »UNIX«, »Quatsch« und »Aprilscherz« über jede gut sortierte Suchmaschine mehrfach im Internet zitiert, beispielsweise auch unter folgender URL:

<http://www.c-plusplus.de/geschichte.htm>

## Taufe

Die Namensgebung von C ist etwas ungewöhnlich, und so spinnen sich Legenden um deren Entstehung. Eine unbestätigte Legende besagt, dass zuerst der Name A im Gespräch war. Aber die Paten fürchteten, dass das A vielleicht in einigen Jahren von Apple patentiert werden würde. Und so ging man das Alphabet durch. B kam nicht in Frage, weil es mit B-Ware oder gar mit BASIC in Verbindung gebracht werden könnte. C hatte dagegen eine freundliche Assoziation. Man könnte Zitronen wählen, um die Lehrbücher über die neue Programmiersprache zu schmücken, und das Gelb würde einfach gut aussehen.

Der Haken an dieser Legende ist, dass die Firma Apple erst ein Jahrzehnt später auf den Plan trat. Außerdem hätten die Namensgeber dann übersehen, dass die Sprache COBOL mit C anfängt. Und das ist sicher eine noch schlimmere Assoziation als BASIC.

Eine deutlich häufiger wiederholte Geschichte besagt, dass man die Sprache erst A nannte, eine spätere Entwicklungsstufe B und dann zu C kam. Und dann hätten sie einfach keine Lust mehr gehabt, die Sprache schon wieder zu verändern. Es war auch ein guter Moment zu stoppen. Die nachfolgenden Buchstaben wären durch D-Zug, E-Mail, F-Wort, G-Punkt und H-Spalterei belegt gewesen. Und wer hätte schon eine Programmiersprache namens I ernst genommen?

Bjarne Stroustrup wusste wohl um die Problematik, sodass er seine Erweiterung der Sprache C um Klassen eben C++ nannte, also ein Inkrement der Sprache C.

## Tempo!

Ganz egal wie es nun zu dem Namen kam. Die Qualitäten der Sprache waren durch das Umfeld gezeichnet. Bis dahin wurden Betriebssysteme in Assembler geschrieben. Dafür gab es vor allem drei Gründe. Assemblerprogramme waren klein, schnell und hatten leichten Zugriff auf die Hardware. Assembler hatte aber auch den Nachteil, dass es das entstehende Betriebssystem zwingend mit einem Prozessor verheiratete. Weitere Nachteile lagen darin, dass die Programme lang und unübersichtlich waren und später kaum wartbar waren. Als UNIX entwickelt wurde, sollte es von der Hardware unabhängig sein und dennoch der Geschwindigkeit von Assembler möglichst nahe kommen.

## Maschinennah

Der Inkrementoperator ist ein Beispiel für diese Optimierung. Fast jeder Prozessor hat einen Inkrementoperator, der schneller ist als die Addition mit 1. Dieser konnte von C direkt angesprochen werden.

Die Zeiger waren von Anfang an so ausgelegt, dass ein direkter Zugriff auf die Controller-Bausteine möglich ist. Controller-Bausteine haben eine feste Speicherstelle

im Adressraum. Weist man diese Adresse der Zeigervariablen direkt zu, kann anschließend der Inhalt des Controllers direkt gelesen und geschrieben werden, und schon war die Verwendung von Assembler zu diesem Zweck unnötig.

C bietet also die Möglichkeit, einen Computer systemnah zu programmieren. Der Zugriff auf Zeiger, das Anfordern und Freigeben von Speicher und der Umgang mit systemnahen Komponenten erfordert eine gewisse Disziplin. Wie in vielen anderen Lebensbereichen ist es auch hier gut, wenn man weiß, was man tut. Die Idee, dass eine Programmiersprache den Computer vor dem Programmierer beschützt, ist jedenfalls in C nicht implementiert.

### **Moderne Sprachkonzepte**

C war aber mehr als ein portabler Assembler. Die damals mit ALGOL aufgekommene strukturierte Programmierung wurde bereits unterstützt. Es gab Schleifenbefehle, Funktionen mit Parametern und lokale Variablen. Darüber hinaus konnten Datenstrukturen modelliert werden, was in jenen Tagen nicht selbstverständlich war.

In jenen Tagen gab es einen erbitterten Kleinkrieg zwischen den Verfechtern der klassischen Programmiersprachen wie COBOL, FORTRAN und BASIC und der aufkommenden strukturierten Programmierung. Die alten Programmiersprachen verwendeten ausgiebig den Befehl GOTO, was dazu führte, dass es sehr schwierig war, dem Source-Code zu folgen, weil die logischen Stränge so durcheinander waren wie die Spaghettis auf einem Teller.

Die Sprache C stammte ähnlich wie PASCAL aus der ALGOL-Linie, die Mechanismen bot, die diesen Spaghetti-Code vermeiden konnte. Allerdings gibt es Programmierer, die mit jeder Sprache in FORTRAN programmieren können. Ein etwas umfangreicherer Artikel, der leicht ironisch die damalige Diskussion widerspiegelt heißt »Echte Programmierer meiden Pascal« und ist vollständig unter folgender URL zu finden:

*<http://www.leo.org/information/freizeit/fun/pascal.html>*

Wer es schätzt, Dinge in der Originalsprache zu lesen, findet eine englische Version unter der folgenden URL:

*<http://www.crusoe.de/pascal.htm>*

Interessanterweise gab es zu jener Zeit auch heftigen Streit zwischen den C- und den PASCAL-Programmierern, welche Sprache die beste sei. Oft fanden sich die Wortgefechte in den einschlägigen Newsgroups. So manch einer Diskussion merkte man an, dass Programmierer wohl nicht so oft wie andere Menschen ihre diesbezüglichen animalischen Veranlagungen zu Stadiengesängen bei der Bundesliga auslebten.

# Index

## A

---

Adjazenzmatrix 48, 54  
ALGOL 17  
Amstrad CPC 340  
Apple  
    *Apple II* 87, 331  
    *Macintosh* 346  
Aprilscherz 14  
ASCII 30  
Assembler 16  
Atari 168  
    *Atari 400 und 800* 335  
    *Atari ST* 348  
Avalanche 241  
    *Spielregeln* 254

## B

---

Bermuda 208  
Bildschirmflackern 236  
Binärbaum 37  
Bloodshed Dev C++ 315  
Busy Waiting 180

## C

---

CeBit 139  
Commodore 168  
    *Amiga* 350  
    *C-64* 338  
CP/M 167

## D

---

deque 249  
Diagonalen 224  
Digital Research Inc. 167  
Dijkstra, Edsger Wybe 45, 170

## E

---

Eichhörnchen 255  
Eliza 21  
Entity Relationship Modell 25  
Epson HX-20 342

Esoterik 151

## F

---

Fibonacci 115  
FIFO 247, 249  
Fließkommakodierung 242  
Fluglandung 167  
FORTRAN 17, 70  
Forum 14  
Freier Fall 64  
Frogger 255

## G

---

Galaxis 207  
Gates, Bill 167  
GIMP 231

## H

---

Hessisch 89

## I

---

IBM 167  
    *PC* 344

## J

---

Jelzin, Boris 207  
Jobs, Steven 139, 331

## K

---

Kaugummispucken 74  
KDevelop 311  
Kernighan, Brian 14  
Kildall, Gary 167  
Kirk, James T. 30

## L

---

Labyrinth 107

## M

---

Macintosh 324  
Marketing 139  
Miner, Jay 168  
Mondlandung 64

## N

---

Nightdriver 191

## O

---

Olivetti P652 63

## P

---

PASCAL 17, 92  
Patterson, Tim 167  
Polling 180  
Portabilitätsprobleme 237  
PortAudio 327  
Pygmalion 22

## R

---

Rekursion 38  
Relationale Datenbank 25  
Ritchie, Dennis 14

## S

---

Schneider CPC 340  
Scotty 21  
Shaw, George Bernhard 22  
Shivji, Shiraz 169  
Sinclair ZX80 337  
Singleton 55, 201  
Spock, Mister 30  
STL  
    *deque* 249  
    *map* 49  
    *vector* 28, 47  
string  
    *find* 34  
    *replace* 34

Superprogramm 139

## T

---

Threads 281  
Tiere raten 36  
Timer 180  
Tramiel, Jack 168  
Twain, Mark 42

## U

---

Umlaute 30  
UNICODE 30  
UTF-8 31

## V

---

Visual Studio 322

## W

---

wchar\_t 31  
Weizenbaum, Joseph 21  
Win32-API 183  
Wozniak, Steve 87, 88, 331  
wxWidgets 139  
    *Bloodshed Dev C++* 320  
    *Ereignis Fenstervergrößerung* 147  
    *Fokus* 260  
    *Installation* 319  
    *KDevelop* 319  
    *Mac* 324  
    *Mausereignis* 148, 159, 197  
    *Maustasten* 221  
    *Menüpunkt mit Haken* 230  
    *Menüs* 144, 157, 230  
    *Messagebox* 145  
    *Sounds* 280  
    *Tastaturereignis* 273  
    *Timer* 180  
    *Visual Studio* 322

## X

---

Xerox 137